That’s awesome! React + Spring Boot + MongoDB Atlas is a powerful stack. Since you’re new to Spring Boot, I’ll guide you step-by-step on how to start the backend part in VS Code. You’ll be building a **Spring Boot application** with **MongoDB** as the database.

**✅ Step 1: Install Prerequisites**

Make sure you have the following installed:

* **Java 17 or above** – [Download](https://adoptium.net/)
* **Maven** – Build tool (comes with many Java IDE setups)
* **VS Code** – with extensions:
  + Java Extension Pack (by Microsoft)
  + Spring Boot Extension Pack (optional but helpful)
* **MongoDB Atlas** – Create a free cluster and get your connection URI

**✅ Step 2: Generate a Spring Boot Project**

Go to: <https://start.spring.io/>

Set these options:

* **Project**: Maven
* **Language**: Java
* **Spring Boot version**: Latest Stable
* **Project Metadata**:
  + Group: com.example
  + Artifact: backend
* **Dependencies**:
  + Spring Web
  + Spring Data MongoDB
  + Spring Boot DevTools (optional for live reload)
  + Lombok (optional for boilerplate reduction)

Click "Generate" to download the project, then unzip it and open the folder in **VS Code**.

**✅ Step 3: Configure MongoDB Connection**

In src/main/resources/application.properties, add:

properties

CopyEdit

spring.data.mongodb.uri=mongodb+srv://<username>:<password>@cluster0.mongodb.net/<dbname>?retryWrites=true&w=majority

spring.data.mongodb.database=<dbname>

server.port=5000

Replace <username>, <password>, and <dbname> with your MongoDB Atlas values.

**✅ Step 4: Create a Simple Model, Repository, and Controller**

Let’s say you want to manage "Users".

**✅ Model (User.java)**

java

CopyEdit

package com.example.backend.model;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "users")

public class User {

@Id

private String id;

private String name;

private String email;

// Constructors, Getters, Setters

}

**✅ Repository (UserRepository.java)**

java

CopyEdit

package com.example.backend.repository;

import com.example.backend.model.User;

import org.springframework.data.mongodb.repository.MongoRepository;

public interface UserRepository extends MongoRepository<User, String> {

}

**✅ Controller (UserController.java)**

java

CopyEdit

package com.example.backend.controller;

import com.example.backend.model.User;

import com.example.backend.repository.UserRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/users")

@CrossOrigin(origins = "\*") // Allow frontend access

public class UserController {

@Autowired

private UserRepository userRepository;

@GetMapping

public List<User> getAllUsers() {

return userRepository.findAll();

}

@PostMapping

public User createUser(@RequestBody User user) {

return userRepository.save(user);

}

}

**✅ Step 5: Run the App**

In VS Code terminal:

bash

CopyEdit

./mvnw spring-boot:run

You should see Spring Boot start on port 5000. Test the API at:

* GET http://localhost:5000/api/users
* POST http://localhost:5000/api/users

**✅ Step 6: Connect Frontend (React)**

In your React app, call the backend endpoints like:

js

CopyEdit

fetch("http://localhost:5000/api/users")

.then(res => res.json())

.then(data => console.log(data));

Perfect! Let’s walk through **Step 1: Installing the necessary tools** for Spring Boot development using **VS Code**.

**✅ Step 1: Install Prerequisites**

Here’s what you need to install:

**🧰 1. Java (JDK 17 or above)**

**👉 Install Java:**

Go to <https://adoptium.net>

* Choose **Temurin 17 (LTS)**
* Download and install it

**✅ After installing:**

Open a terminal and check:

bash

CopyEdit

java -version

You should see something like:

nginx

CopyEdit

openjdk version "17.0.x" ...

**🧰 2. Maven (Build tool used by Spring Boot)**

**👉 Install Maven:**

* Download from <https://maven.apache.org/download.cgi>
* Extract it
* Add bin folder to your system’s PATH

**✅ Check Maven version:**

bash

CopyEdit

mvn -v

You should see:

nginx

CopyEdit

Apache Maven 3.x.x

**🧰 3. VS Code (IDE)**

If you haven't already, download from:  
<https://code.visualstudio.com>

**🧩 4. Install VS Code Extensions**

In VS Code, install the following extensions:

1. **Java Extension Pack** by Microsoft  
   *(Includes everything needed: Language support, IntelliSense, debugger, etc.)*
2. **Spring Boot Extension Pack** *(Optional but helpful for Spring apps)*

To install:

* Open VS Code
* Go to **Extensions (Ctrl+Shift+X)**
* Search for **Java Extension Pack** and click Install

**🧪 5. MongoDB Atlas Setup**

If you haven’t already:

* Go to <https://www.mongodb.com/cloud/atlas>
* Create a free cluster
* Set up a **Database User**
* Whitelist your IP (or allow all 0.0.0.0/0 for development)
* Copy the connection string (MongoDB URI)

**✅ Once all this is done:**

You're ready to move to **Step 2: Generate the Spring Boot project**.

**So the core files to work with are:**

**mvn spring-boot:run**

pom.xml

src/main/java/com/example/demo/DemoApplication.java

src/main/resources/application.properties

**✅ Important Files & Folders (You Should Know)**

**📄 pom.xml**

* **Most important config file** for your project.
* Declares dependencies (like Spring Web, Spring Data MongoDB, etc.).
* You’ll edit this when you add new libraries (e.g., JWT, Lombok, etc.).

**📁 src/main/java/com/example/demo/**

**📄 DemoApplication.java**

* This is your main class with the main() method.
* Starts the Spring Boot app with @SpringBootApplication.
* Later, this is where you might enable Mongo auditing, etc.

**📁 src/main/resources/**

**📄 application.properties**

* Used to configure your application (e.g., MongoDB URI, port, JWT secrets).
* You’ll set things like:

properties

CopyEdit

spring.data.mongodb.uri=mongodb+srv://your-user:pass@cluster.mongodb.net/db

server.port=8080

**✅ Suggested Folder Structure**

src/main/java/com/example/backend/

├── model/

│ └── Post.java

├── controller/

│ └── PostController.java

├── service/

│ └── PostService.java

├── repository/

│ └── PostRepository.java

└── config/

src/main/resources/

└── application.properties

uploads/

└── (this is where uploaded images/videos will be stored)

**📁 What Each Folder Does**

| **Folder** | **Purpose** |
| --- | --- |
| controller/ | REST API endpoints. Example: UserController.java |
| model/ | MongoDB data models using @Document. Example: User.java |
| repository/ | Interfaces for MongoDB operations. Example: UserRepository.java |
| service/ | Business logic layer. Keeps controllers clean. |
| config/ | For config files like MongoDB, CORS, etc. |
| resources/application.properties | MongoDB URI, server port, etc. |
| DemoApplication.java | Entry point — runs your app. |

**🔁 When you send this request:**

http

CopyEdit

POST http://localhost:8080/api/users

Content-Type: application/json

{

"name": "Dalton",

"email": "dalton@example.com"

}

**✅ Step-by-step flow:**

1. **🔗 The Request hits the Controller**
   * The Spring Boot app is listening at /api/users.
   * The @PostMapping method in UserController is triggered.
   * It automatically **maps** the incoming JSON to a User object.

java

CopyEdit

public User createUser(@RequestBody User user)

Now you have: user.name = "Dalton", user.email = "dalton@example.com".

1. **📞 Controller calls the Service**
   * Inside the controller, this line is called:

java

CopyEdit

return userService.createUser(user);

* + Now control moves to UserService.

1. **💾 Service calls the Repository**
   * Inside UserService, it calls:

java

CopyEdit

return userRepository.save(user);

* + This uses Spring Data MongoDB (via MongoRepository) to **store the user in MongoDB Atlas**.

1. **📚 MongoDB stores the user**
   * MongoDB receives the data and automatically generates a unique \_id (unless you specify it).
   * It saves the user in the **users collection** in your MongoDB database.
2. **📤 Response goes back**
   * The saved user object (with the generated ID) is returned from the repository → service → controller → client.
   * You'll receive a response like:

json

CopyEdit

{

"id": "6626869c75a9981234567890",

"name": "Dalton",

"email": "dalton@example.com"

}

**📌 In short:**

arduino

CopyEdit

Client → Controller → Service → Repository → MongoDB → Back to Client

It’s like:

"Hey backend, here’s a new user → Save it in the DB → Done! Here’s the saved user info back."

Let me know if you want to add validation (like checking for duplicate emails), or want to test it with Postman or React next!